Pemrograman pada MATLAB

5.1 Struktur dan Tipe Data

Sebelum membahas tentang pemrograman, akan lebih baik jika kita mengetahui tentang struktur data dan tipenya dalam MATLAB. Tipe data yang digunakan pada MATLAB standarnya adalah tipe data double. Tetapi, tipe data tersebut dapat berubah menyesuaikan dengan nilai yang diberikan kepadanya. Beberapa tipe data yang juga merupakan fungsi pengubah tipe data, terdapat pada Tabel 5.1.

Tabel 5.1 Tipe Data pada MATLAB

Tipe Data	Keterangan
single	bilangan pecahan presisi tunggal
double	bilangan pecahan presisi ganda
int8, int16, int32, int 64	bilangan bulat 8, 16 , 32, atau 64 bit bertanda
uint8, int16, uint32, uint64	bilangan bulat 8, 16, 32 atau 64 bit tak bertanda. (tipe data yang tidak menerima bilangan negative)
char	Tipe data yang berupa karakter atau string

Sedangkan untuk struktur data MATLAB terdapat pada Tabel 5.2 dan untuk menampilkan format bilangan terdapat pada Tabel 5.3.

Tabel 5.2 Struktur Data pada MATLAB

Struktur Data	Keterangan
Multidimensional arrays	array dengan tiga atau lebih subscript. Dapat dibentuk dengan memanggil fungsi zeros, ones, rand, atau randn dengan argument lebih dari dua.
Cell arrays	multidimensional arrays yang elemen-elemennya dikopi dari array yang lain. Cell array kosong dapat dibentuk dengan fungsi cell. Cell array dibentuk dengan melingkupi kumpulan suatu data dengan kurung kurawal. '{ } '. Kurung kurawal juga digunakan untuk mengakses isi dari berbagai sel.
Characters and text	teks yang diawali dan diakhiri dengan apostrof ('). Setiap karakter dalam suatu string adalah satu elemen array, dengan setiap elemennya sebesar 2 byte.
Structure	Kumpulan dari array yang tersusun menurut field dan valuenya.

Tabel 5.3 Format Tampilan Bilangan

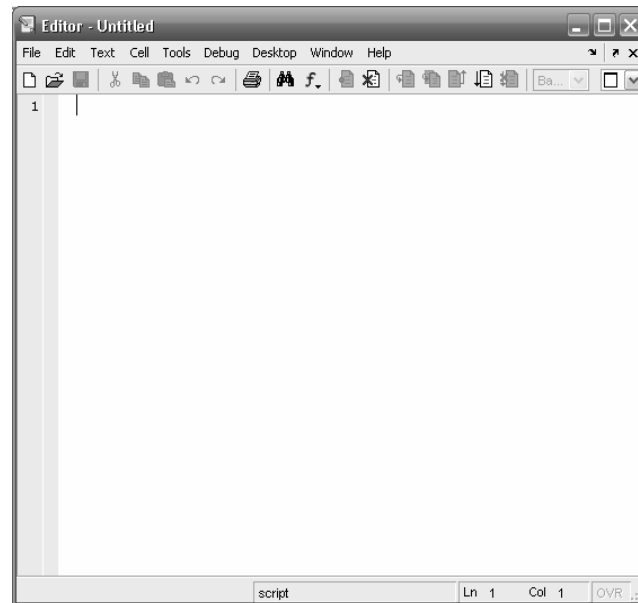
Simbol	Keterangan
%d	Bilangan bulat
%f	Bilangan pecahan
%o	Bilangan okta
%x, %X	Bilangan hexadesimal
%e, %E	Bilangan sebagai a x 10^b
/n	Pindah ke baris baru
/t	Geser sepanjang 1 tab

Tampilan bilangan %f, %e dan %E, ketepatan dapat diatur dengan menyisipkan format presisi bilangan. Format presisi adalah .p, dengan p berupa nilai yang menyatakan banyaknya angka di belakang koma. Lihat contoh berikut ini.

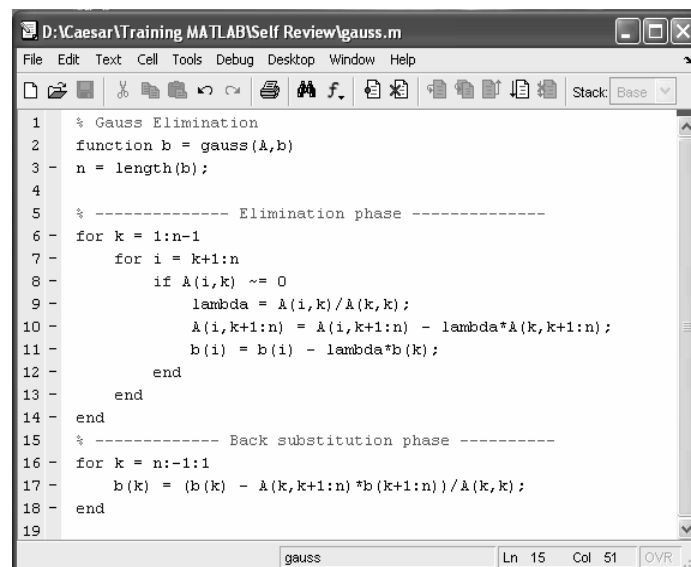
```
>> a = input ('Bilangn I = ');
Bilangan I = 3

>> b = input ('Bilangan II =');
Bilangan II =4

>> c = a*b;
>> fprintf ('%f dikali %f sama dengan %f \n', a, b, c);
3.000000 dikali 4.000000 sama dengan 12.000000

>> fprintf ('%.3f dikali %.3f sama dengan %.3f \n', a, b, c);
3.000 dikali 4.000 sama dengan 12.000

>> fprintf ('%e dikali %e sama dengan %e \n', a, b, c);
3.000000e+000 dikali 4.000000e+000 sama dengan 1.200000e+001

>> fprintf ('%.3e dikali %.3e sama dengan %.3e \n', a, b, c);
3.000e+000 dikali 4.000e+000 sama dengan 1.200e+001
```

5.2 M-File

Kita sudah mempelajari penulisan variabel dan perhitungan operasi-operasi matematika di Command Window. Coba bayangkan jika kita harus menulis suatu pemrograman yang sangat panjang di dalam Command Window, pasti akan sangat merepotkan. Anda harus mengetik suatu perhitungan di MATLAB setiap Anda membutuhkan, atau jika Anda melakukan kesalahan akan sangat sulit untuk menghapusnya. Kelemahan yang paling utama adalah program yang Anda buat tidak dapat disimpan di dalam komputer. Untuk mengakomodasi masalah ini, MATLAB memiliki wadah untuk melakukan pemrograman yang bernama M-File. Di dalam M-File, program Anda dapat disimpan di dalam komputer dan bisa Anda buka dan gunakan kapan pun Anda inginkan. Terdapat tombol khusus jika Anda ingin mengeksekusi program Anda.

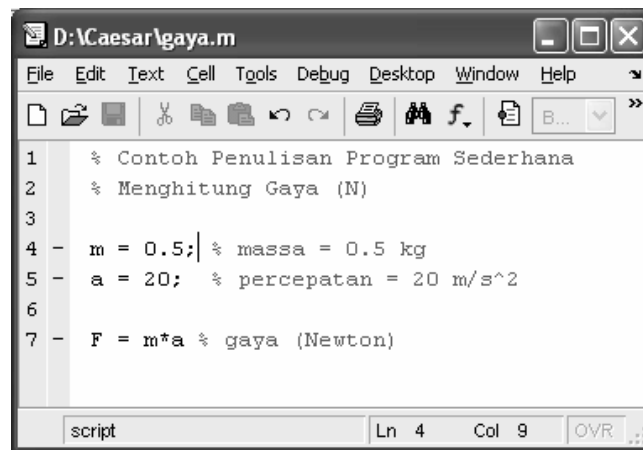
Untuk memulai skrip M-File, Anda bisa memulai dengan membuka file baru melalui main window **File → New → M-File** atau mengklik ikon yang ada di jendela utama. Setelah itu akan terbuka jendela seperti di bawah ini.

Gambar 5.1 M-File yang belum ditulis program

D:\Caesar\Training MATLAB\Self Review\gauss.m

```
1    % Gauss Elimination
2    function b = gauss(A,b)
3 -  n = length(b);
4
5    % -------------- Elimination phase --------------
6 -  for k = 1:n-1
7 -      for i = k+1:n
8 -          if A(i,k) ~= 0
9 -              lambda = A(i,k)/A(k,k);
10 -             A(i,k+1:n) = A(i,k+1:n) - lambda*A(k,k+1:n);
11 -             b(i) = b(i) - lambda*b(k);
12 -         end
13 -     end
14 - end
15   % ------------- Back substitution phase ----------
16 - for k = n:-1:1
17 -     b(k) = (b(k) - A(k,k+1:n)*b(k+1:n))/A(k,k);
18 - end
19
```

Gambar 5.2 M-File yang telah ditulis program

5.3 M-File untuk Membuat Skrip Sederhana

Sebagai contoh, kita ingin menghitung besaran gaya dengan variabel yang diketahui adalah massa dan percepatan.

```
D:\Caesar\gaya.m
File Edit Text Cell Tools Debug Desktop Window Help

1     % Contoh Penulisan Program Sederhana
2     % Menghitung Gaya (N)
3
4 -   m = 0.5; % massa = 0.5 kg
5 -   a = 20;  % percepatan = 20 m/s^2
6
7 -   F = m*a % gaya (Newton)

script                Ln 4    Col 9    OVR
```

Gambar 5.3 Contoh m-file sederhana

Setelah program dieksekusi besaran gaya (F) dengan satuan Newton akan tampil pada Command Window.

```
F =
    10
```

5.4 M-File untuk Membuat Fungsi

Matlab memiliki banyak fungsi yang sudah dibuat seperti contoh yang telah disampaikan pada bab sebelumnya, yaitu fungsi determinan. Kita juga bisa membuat fungsi sesuai dengan kebutuhan. Sebagai contoh, Anda ingin membuat fungsi luas lingkaran atau jika ingin lebih rumit lagi, Anda ingin membuat fungsi eliminasi Gauss seperti yang telah ditampilkan pada gambar M-File. Dengan membuat fungsi sendiri, Anda akan bisa mengeksekusi sesuai dengan nilai masukan yang Anda inginkan.

Bentuk penulisan fungsi adalah:

```
function [nilai_keluaran] = nama_fungsi(nilai_masukan)
        isi fungsi atau persamaan matematika
end
```

Sebagai contoh, kita ingin membuat fungsi *force* (gaya) dengan langkah-langkah sebagai berikut:

1. Buka M-File baru.
2. Ketik sintak di bawah ini dalam M-File.

```
function [F] = gaya(m,a)
    F = m*a
end
```

3. Simpan sesuai dengan nama fungsi (dalam hal ini gaya.m).
4. Ketik perintah di bawah ini dalam Command Window dan berikan nilai m dan a (contoh m = 1 dan a = 20). Maka akan dihasilkan jawaban F = 20.

```
>> [F] = gaya(1,20)
F =
    20
```

5.5 Logical Statements

Matlab bisa digunakan untuk pemrograman yang bersifat *logical* dengan mengaplikasikan salah satu dari pemrograman *logic* atau bisa juga dengan cara menggabungkan atau mengombinasi dari dua atau lebih pemrograman logika tersebut. Umumnya *Logical Statements* dibagi menjadi dua berdasarkan kendali aliran program, yaitu percabangan bersyarat dan kendali perulangan. *If statements* dan *Switch Statements* termasuk percabangan bersyarat, sedangkan *For Loop* dan *While Statements* termasuk kendali perulangan.

5.5.1 If Statements

Pemrograman ini merupakan statement untuk percabangan program berdasarkan satu/beberapa kondisi/syarat tertentu. Sintaks yang digunakan dalam MATLAB terdapat pada Tabel 5.4.

Tabel 5.4 Syntax If Statements

Satu syarat	Dua syarat	Tiga syarat atau lebih
if syarat Dijalankan jika syarat dipenuhi; **end**;	**if** syarat Dijalankan jika syarat dipenuhi; **else** Dijalankan jika syarat tidak dipenuhi; **end**;	**if** syarat1 Dijalankan jika syarat1 dipenuhi; **elseif** syarat 2 Dijalankan jika syarat 2 dipenuhi; **elseif** syarat 3 Dijalankan jika syarat 3 dipenuhi; **elseif**.... .... **else**.... Dijalankan jika syarat selain di atas tidak dipenuhi;; **end**;

Sebagai contoh, ketikkan code berikut pada m-file.

```
% klasifikasi skor toefl
Toefl=558
If       (Toefl >= 601) & (Toefl <= 670)
         display('Excellent')
elseif Toefl >= 501) & (Toefl <= 600)
       display('Good')
elseif (Toefl >= 401) & (Toefl <= 500)
       display('Average')
else   Toefl<=400
       display('bad')
end
```

Contoh tersebut menggunakan operator logika seperti yang tertulis pada Tabel 5.4. Setelah m-file tersebut dijalankan, akan terlihat jawaban sebagai berikut.

```
Toefl =
   558
Good
```

Sekarang kita coba contoh kedua menggunakan masukan 'input'.

```
a=input('masukan nilai a=')
if a > 0
    y = 2*a+5
    display('a=positif')
elseif a == 0
    y = a-6
    display('a = nol')
else a < 0
    y = 2+a
    display('a =negatif')
end
```

```
masukkan nilai a=5
a =
     5
y =
    15
a =positif
```

5.5.2 Switch Statements

Selain menggunakan pemrograman bercabang dengan *if statement*, kita dapat melakukkannya menggunakan *switch statement* yang mempunyai *syntax* hampir sama dengan *if statement,* seperti berikut ini.

```
switch syarat
        case 1
        pernyataan 1
        case 2
        pernyataan 2
        ...
       otherwise
       pernyataan N
end;
```

Sebagai contoh, kita akan membagi suatu bilangan tertentu dengan angka 7, kemudian buat program yang akan menampilkan sisa hasil pembagian tersebut.

```
a=input('a='); % masukan nilai input a
n=mod(a,7);    % a dibagi oleh 7
switch n
    case 0
     disp('tak bersisa')
    case 1
     disp('sisa satu');
    case 2
     disp('sisa dua');
    case 3
     disp('sisa tiga');
    case 4
     disp('sisa empat');
    case 5
     disp('sisa lima');
    otherwise
     disp('sisa enam');
end;
```

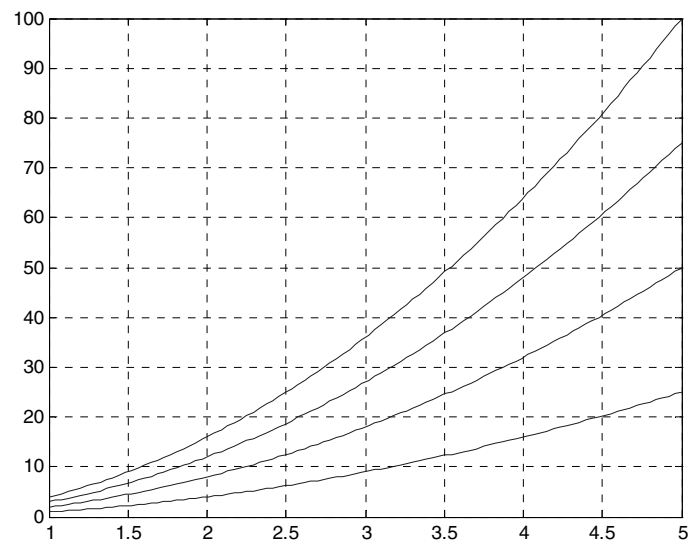
```
a=608
sisa enam
```

5.5.1 For Loop

Terdapat dua macam perulangan, yaitu perulangan berbatas *for* dan perulangan bersyarat *while*. *For loop* digunakan jika syarat nilai akhir dari perulangan dapat ditentukan dengan pasti. Di bawah ini adalah format pemrograman menggunakan sintak for.

```
for variabel = nilai_awal : inkremen : nilai_akhir
Command untuk dijalankan
end  % berhenti jika sudah mencapai nilai_akhir.
```

Contoh:

```
t = linspace(1,5,100); % mendefinisikan data t dimulai dari
angka 1 dan diakhiri angka 5 dengan jumlah data sebanyak 100
A = 1:1:4;  % mendefinisikan vektor A (1 2 3 4)
for i = 1:length(A)
 y = A(i)* t.^2;
 plot(t,y);
 hold on;
end
grid on;
```

5.5.4 While Statements

Syarat nilai akhir dari perulangan tidak bisa ditentukan dengan pasti. Untuk itu, digunakan perulangan bersyarat menggunakan *while statement*. Jika nilai pencacah memenuhi syarat perulangan, maka perulangan dilanjutkan. Jika nilai pencacah tidak memenuhi syarat perulangan, maka perulangan dihentikan.

```
while kondisi
         Command untuk dijalankan jika kondisi dipenuhi
end  % keluar dari loop jika kondisi tidak dipenuhi
```

```
n=0;               % nilai awal n=0
jumlah=0;
while (jumlah <= 100) % syarat perulangan
   jumlah=jumlah + n^2-3*n;
   n = n+2;     % pertambahan positif bilangan bulat
end;
fprintf('total = %.3f dengan n = %d \n', jumlah, n);
```

Total = 130.000 dengan n = 12

```
% tegangan referensi keluaran
vr = input ('referensi keluaran (volt) = ');

% gelombang sinusoidal
frek = input ('frekuensi (hz) = ');
t = (0 : 0.01 : 0.5); % waktu selama 0.5 detik
vin = cos(2 * pi * frek * t);

% Faktor pembanding
n = 1;
while (n <= length (vin))
   if (vin(n) < 0)
      vout (n) = -1 * vr;
   elseif (vin(n) < 0)
      vout (n) = vr;
   else
      vout (n) = 0;
   end;
   n = n + 1;
end;

% membuat plot waktu dengan tegangan masuk dan tegangan
keluar
plot ( t, [vin; vout]); % membuat plot grafik
legend ('vin', 'vout'); % memberikan legenda
```

Referensi keluaran (volt) = 10

Frekuensi (hz) = 5